***Текст програми мовою С***

***main.cpp***

#include <unistd.h>

#include <semaphore.h>

#include <pthread.h>

#include <fstream>

#define MAX\_BUFFER\_SIZE 10

using namespace std;

ofstream cout("out.log");

pthread\_t thread\_1;

pthread\_t thread\_2;

pthread\_t thread\_3;

pthread\_t thread\_4;

pthread\_t thread\_5;

sem\_t SCR1, sem\_1, sem\_2;

pthread\_mutex\_t MCR1 = PTHREAD\_MUTEX\_INITIALIZER;

int data = 0;

int common\_resource[MAX\_BUFFER\_SIZE];

int next\_in = 0, next\_out = 0;

int passageway = 0;

void\* producer(void \*\_id){

int id = \*(int\*)\_id;

int sem\_value;

while(true){

if(id == 5){

sem\_post(&sem\_2);

sem\_wait(&sem\_1);

}

sem\_getvalue(&SCR1, &sem\_value);

//if (sem\_value == 0) break;

if(passageway == 0 && sem\_value == MAX\_BUFFER\_SIZE) passageway++;

else if(passageway == 1 && sem\_value == 0) passageway++;

else if(passageway == 2 && sem\_value == MAX\_BUFFER\_SIZE) passageway++;

else if(passageway == 3 && sem\_value == 0) break;

if(sem\_value < MAX\_BUFFER\_SIZE)

if (pthread\_mutex\_trylock(&MCR1) == 0){

common\_resource[next\_in] = data++;

next\_in = (next\_in + 1) % MAX\_BUFFER\_SIZE;

cout << "Producer thread " << id << ": semaphore = " << sem\_value+1 << "; element " << data - 1 << " CREATED;\n";

pthread\_mutex\_unlock(&MCR1);

sem\_post(&SCR1);

}

else {

//cout << "Producer thread " << id << " does some useful work\n";

}

}

cout << "Producer thread " << id << " stopped!!!\n";

pthread\_cancel(thread\_2);

pthread\_cancel(thread\_3);

pthread\_cancel(thread\_4);

if (id == 1)

pthread\_cancel(thread\_5);

return NULL;

}

void\* consumer(void \*\_id){

int id = \*(int\*)\_id;

int sem\_value;

while(true){

if(id == 4){

sem\_post(&sem\_1);

sem\_wait(&sem\_2);

}

sem\_getvalue(&SCR1, &sem\_value);

if (passageway == 3 && sem\_value == 0) break;

if(sem\_trywait(&SCR1) == 0)

if( pthread\_mutex\_trylock(&MCR1) == 0) {

sem\_getvalue(&SCR1, &sem\_value);

int value = common\_resource[next\_out];

next\_out = (next\_out + 1) % MAX\_BUFFER\_SIZE;

cout << "Consumer thread " << id << ": semaphore = " << sem\_value << "; element " << value << " DELETED;\n";

pthread\_mutex\_unlock(&MCR1);

}

else {

sem\_post(&SCR1);

//cout << "Consumer thread " << id << " does some useful work\n";

}

else{

//cout << "Consumer thread " << id << " does some useful work\n";

}

usleep(2);

}

cout << "Consumer thread " << id << " stopped!!!\n";

return NULL;

}

int main(){

sem\_init(&SCR1, 0, 0);

sem\_init(&sem\_1, 0, 0);

sem\_init(&sem\_2, 0, 0);

int id\_thread\_1 = 1;

int id\_thread\_2 = 2;

int id\_thread\_3 = 3;

int id\_thread\_4 = 4;

int id\_thread\_5 = 5;

for(; data < 5; ++data) {

common\_resource[data] = data;

sem\_post(&SCR1);

++next\_in;

}

cout << "Queue with elements from 0-th to " << 9 << "-th has been created!!!\n";

int sem\_value;

sem\_getvalue(&SCR1,&sem\_value);

cout << "semaphore = " << sem\_value << endl;

pthread\_create(&thread\_1,NULL,&producer,(void\*)&id\_thread\_1);

pthread\_create(&thread\_2,NULL,&consumer,(void\*)&id\_thread\_2);

pthread\_create(&thread\_3,NULL,&consumer,(void\*)&id\_thread\_3);

pthread\_create(&thread\_4,NULL,&consumer,(void\*)&id\_thread\_4);

pthread\_create(&thread\_5,NULL,&producer,(void\*)&id\_thread\_5);

pthread\_join(thread\_1, NULL);

pthread\_join(thread\_2, NULL);

pthread\_join(thread\_3, NULL);

pthread\_join(thread\_4, NULL);

pthread\_join(thread\_5, NULL);

return 0;

}